VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

INFORMACINIŲ TECHNOLOGIJŲ KATEDRA

**LABORATORINIS DARBAS 3**

Atliko: DGTfm 22 stud.

Matas Malickas

Priėmė: doc. dr. Andrej Bugajev

Vilnius, 2023

Darbo eiga

Užduoties numerio apskaičiavimas

Pagal formulę ((n − 1) mod 4) + 1, priskyrus n = 8, buvo apskaičiuotas užduoties skaičius: 4.

Užduotis

Ištirkite tiesinę priklausomybę parametro "BendrasLaikas" nuo parametro "svoris" (agreguodami pagal maršrutą ir datą, pritaikykite sumos operaciją), kai nagrinėjami duomenys tik su viena ta pačia reikšme "Masinos tipas". Analizės metu pritaikykite tiesinę regresiją.

Spark konkteksto inicializaivmas

Norint naudotis „Apache Spark“ lokalioje aplinkoje buvo įdiegtas „pyspark“ pip paketas naudojantis komanda:

pip install pyspark

Pirmojoje Jupyner Notebook celėje buvo inicializuotas „Apache Spark“ kontekstas, kad būtų galima atlikti tolimesnius veiksmus su RDD duomenų struktūromis. Taip pat užkraunamas tekstinis failas su duomenimis.

from pyspark import SparkConf, SparkContext

conf = SparkConf().setAppName('MyApp')

sc = SparkContext(conf=conf)

A = sc.textFile('duom\_cut.txt')

Reikšmių išskyrimas iš failo ir mapinimas

Toliau yra apdorojamas duomenų failas naudojant map funckijas, kurios sudaro objektų sąrašus ir panaikina atskyrimo simbolius. Padarius tvarkingus duomenis, mapinama per objekto raktus ir reikšmes, ir jei raktas atitinka „svorio grupe“, tai to įrašo vertė priskiriama kaip raktas, o jei atrandamas įrašas su raktu „svoris“, jo vertė priskiriama kaip mapinimo funkcijos gražinama reikšmė.

def map\_func(x):

key = ''

value = float(0)

for entry in x:

e\_key, e\_value = entry.split('=')

if e\_key == 'svorio grupe':

key = e\_value.replace('}}', '').replace('{{', '')

elif e\_key == 'svoris':

try:

value = float(e\_value.replace('}}', '').replace('{{', ''))

except Exception as e:

pass

return (key, value)

y = A.map(lambda x: x.strip())\

.flatMap(lambda x: x.split('}}{{'))\

.map(lambda x: x[2:] if x.startswith('{{') else x)\

.map(lambda x: x[:-2] if x.startswith('}}') else x)\

.map(lambda x: x.split('}{'))\

.map(map\_func)\

.filter(lambda x: x[0] != '')

print(y.take(20))

Šioje būsenoje duomenys yra paruošti reduce operacijai.

Reduce

Siekiant apskaičiuoti mažiausią, didžiausią bei vidutinius vidurkius, buvo atliekama reduce operacija. Iteruojant per visas vertes, buvo sumapinami tuple tipo objektai, kurių pirma vertė yra svoris, o antra vertė yra 1, kuris bus naudojamas sekti sudėtų kintamųjų kiekiui.

,,reduce\_func” funkcija yra paduodama Spark funkcijai reduceByKey ir ji sudeda svorių reikšmes bei jų pasikartojimo reikšmes bei gražina tuple tipo objektą.

def reduce\_func(x, y):

return (x[0] + y[0], x[1] + y[1])

reduced = y.mapValues(lambda x: (x, 1))\

.reduceByKey(reduce\_func)

print(reduced.take(20))

Gautas rezultatas su nepilnais duomenimis

Siekiant paskaičiuoti statistikas, buvo įvykdytos mapinimo funkcijos, kurios apskaičiuoja vidurkius, minimalias ir maksimalias reikšmes kiekvienai svorių grupei.

avg = reduced.mapValues(lambda x: x[0] / x[1])

print(avg.take(20))

min = reduced.mapValues(lambda x: x[0] if x[0] < x[1] else x[1])

print(min.take(20))

max = reduced.mapValues(lambda x: x[0] if x[0] > x[1] else x[1])

print(max.take(20))

Įvykdžius šias operacijas buvo gauti tokie rezultatai:

[('>300', 516.7904761904762), ('<300', 100.26420454545455), ('<50', 10.383368421052634)]

[('>300', 21), ('<300', 176), ('<50', 285)]

[('>300', 10852.599999999999), ('<300', 17646.5), ('<50', 2959.2600000000007)]